**Team "Ascendance"**

Team members:

**Любомир Йончев**

**Иван Дойчинов**

**Любомир Свиленов - retired**

Project:   
*Santace semi-smart bot* ☺

### Project Description

A bot with implemented logic for playing Santase game

* Based on Minimax algorithm

The URL of our source control repository is:

<https://idoychinov.visualstudio.com/DefaultCollection/AscendanceAI/AscendanceAI%20Team/_git/AscendanceAI>

### Algorithm and details explanation

* The bots logic is based on the Minmax algorithm, [Link](https://en.wikipedia.org/wiki/Minimax)
* Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision making in the presence of uncertainty.
* Basically on every move the bot takes pool of all possible cards to be play from the opponent and make a weight for every combination between opponentPossibelCards and his cards. Then he always play that card which will provide the highest weight possible value.

Steps:

* We have **OpponentPosibleCardsProvider** method which calculate the pool of the opponent card on every turn. It’s a basic formula which takes allCards minus trump card and playedCards.
* In **Chose Action Method** the opponentPossibleCards has been taken via Helper - OpponentPosibleCardsProvider. For each card in possibleCardsToPlay we calculate the weight and stored them in Dictionary with a key Card, and value Weight.  
  Then we play the card with the maximum possible value.
* The Weight calculation is in **CalculateCardPlayedWeight** method.  
  Basically we took the value of both played card in every turn and sum them. If the value of the opponent card is lower then ours the value is with “+” sign. In the other case is with “-“. It’s a serious if else logic so pay attention ☺
* In **ShouldCloseGame** method we implement a logic for game closures with a several conditions.
* We have additional logic for the 20 and 40 announcements in the method **TryToAnnounce20Or40.** It similar to the smart player one.